Environmentally Transmitted Pathogens

Models – Part deux :)

Julien Arino

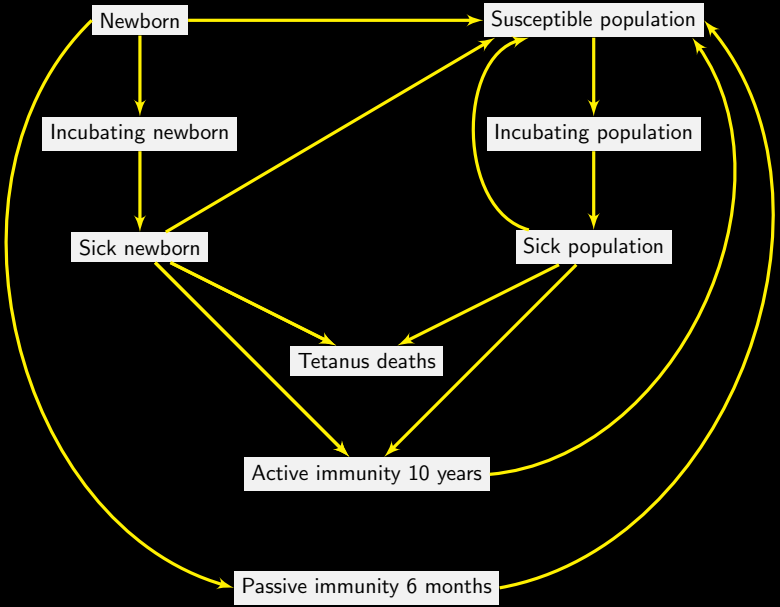
January 2023

Some considerations about numerics

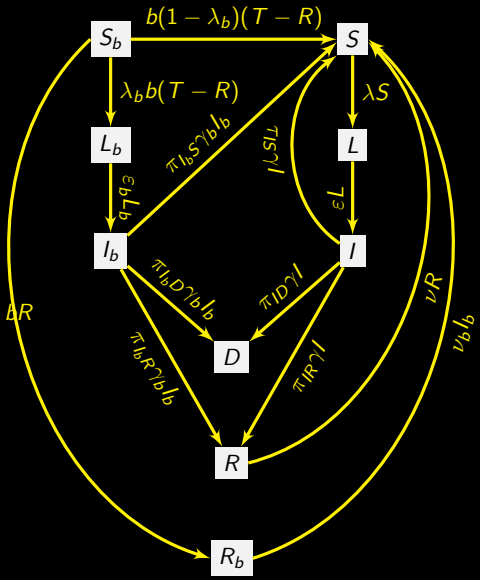
- The tetanus model of Cvjetanović
- The model of Capasso for ETP
- The first schistosomiasis model of Woolhouse
- The third schistosomiasis model of Woolhouse – Heterogeneous contacts

Some considerations about numerics

- The tetanus model of Cvjetanović
- The model of Capasso for ETP
- The first schistosomiasis model of Woolhouse
- The third schistosomiasis model of Woolhouse – Heterogeneous contacts


Newborn
Susceptible population
Incubating newborn
Incubating population
Sick newborn
Sick population
Tetanus deaths
Active immunity 10 years
Passive immunity 6 months

Flow diagram (demography not shown)



The discrete-time tetanus model (notation mine)

$$\Delta S_b = bT \tag{1a}$$

$$\Delta S = b(1-\lambda_b)(T-R) + \nu R + \nu_b I_b + \nu I + \pi_{I_b S}\gamma_b I_b + \pi_{IS}\gamma I - (\lambda + d - \delta_T)S \tag{1b}$$

$$\Delta L_b = \lambda_b b(T-R) - (\varepsilon_b + d - \delta_T)L_b \tag{1c}$$

$$\Delta L = \lambda S - (\varepsilon + d - \delta_T)L \tag{1d}$$

$$\Delta I_b = \varepsilon_b L_b - (\gamma_b + d - \delta_T)I \tag{1e}$$

$$\Delta I = \varepsilon L - (\gamma + d - \delta_T)I \tag{1f}$$

$$\Delta R = \pi_{I_b R}\gamma_b I_b + \pi_{IR}\gamma I - (\nu + d - \delta_T)R \tag{1g}$$

$$\Delta R_b = bR - (\nu_b + d - \delta_T)R_b \tag{1h}$$

$$\Delta D = \pi_{I_b D}\gamma_b I_b + \pi_{ID}\gamma I \tag{1i}$$

where

$$T = S + L_b + L + I_b + I + R + R_b \quad \text{and} \quad \delta_T = \frac{\Delta D}{T} \tag{1j}$$

Parameter assumptions – Tetanus

- **Incubation period** – Mean duration 6 days for newborn and 8 days for general population $\Rightarrow$ daily rate of exit $\varepsilon_b = 0.1667$ and $\varepsilon = 0.125$
- **Period of sickness** – Mean duration 3 days for newborn and 14 days for general population $\Rightarrow$ daily rate of exit $\gamma_b = 0.3333$ per sick newborn and $\gamma = 0.0714$ for sick general in general population
- **Mortality from tetanus** – Untreated tetanus cases, fatality rate 90% for newborn S_b and 40% for general population. Treated: 80% for newborn and 30% general population
- **Immunity** – Tetanus cases do not lead to immunity to reinfection. But as a general rule, recovered people are vaccinated. Convalescents and general population effectively immunised by complete course of vaccination go to R for average 10 years, daily rate of exit is $\nu = 0.000274$ per person.
- **Immunity of newborns** – Newborn to women vaccinated during pregnancy are temporarily protected by maternal antibodies and pass through R_b for a mean duration of 6 months. Daily rate of exit $\nu_b = 0.005479$ per immunised newborn

Deciding on infection outcome – π

Parameters π are proportion of individuals who follow a certain route post-infection

- $\pi_{I_b\bullet}$ proportion of infected newborn who
  - π_{I_bS} recover without immunity
  - π_{I_bR} recover with immunity
  - π_{I_bD} die (0.9)

  $\pi_{I_bS} + \pi_{I_bR} + \pi_{I_bD} = 1$

- $\pi_{I\bullet}$ proportion of infected who
  - π_{IS} recover without immunity
  - π_{IR} recover with immunity
  - π_{ID} die (0.4)

  $\pi_{IS} + \pi_{IR} + \pi_{ID} = 1$

Parameter assumptions – Demography

Live birth rate 35 per 1,000 population and annual crude death rate 15 per 1,000 population (annual rate of growth 2%) $\Rightarrow$ daily birth and death rates $b = 0.00009889$ and $d = 0.0000411$ per person, respectively

Parameter assumptions – Force of infection

No H2H transmission $\Rightarrow$ incidence proportional to number of susceptible individuals and force of infection, which quantifies combined effect of all variables involved in infection process:

- degree of soil contamination with *Clostridium tetani*
- climate
- frequency of lesions
- proportion of rural population
- socioeconomic conditions
- level of medical care for the wounded and during deliveries

Force of infection acting on newborn (λ_b) and susceptible population (λ) fixed at 3 different levels adequate for reproducing the following stable annual incidence rates of tetanus cases in the community

- For newborn, 200 cases, 400 cases and 600 cases per 100,000 newborn
- For general population (without newborn), 9, 18 and 27 cases

A crash course on discrete-time systems

We have seen systems of ordinary differential equations (ODE) of the form

$$\frac{d}{dt}x(t) = f(x(t))$$

often written omitting dependence on t, i.e.,

$$x' = f(x) \tag{2}$$

where $x \in \mathbb{R}^n$ and $f : \mathbb{R}^n \to \mathbb{R}^n$. The system is considered together with an initial condition $x(t_0) = x_0 \in \mathbb{R}^n$.

The **independent** variable $t \in \mathbb{R}$

A discrete-time system takes the form

$$x(t + \Delta t) = f(x(t)) \tag{3}$$

where $x(t) \in \mathbb{R}^n$ and $f : \mathbb{R}^n \to \mathbb{R}^n$

In a discrete-time system, t is discrete and can be assumed to be in $\mathbb{Z}$ or $\mathbb{N}$ (in practice, before "recasting", it is in $\mathbb{Q}$), we often write $x(t + 1) = f(x(t))$, assuming $\Delta t = 1$..

Together with an initial condition $x(t_0) = x_0 \in \mathbb{R}^n$, this constitutes a sequence that describes the evolution of the state x

Similarities/differences

$$x' = f(x), x(t_0) = x_0, x \in \mathbb{R}^n \qquad x(t + \Delta t) = f(x(t)), x(t_0) = x_0, x \in \mathbb{R}^n$$

Equilibria (EP) $x^\star$ s.t. $f(x^\star) = 0_{\mathbb{R}^n}$ Fixed points (FP) $x^\star$ s.t. $f(x^\star) = x^\star$

LAS EP $\Leftrightarrow s(Df(x^\star)) < 0$ LAS FP $\Leftrightarrow \rho(Df(x^\star)) < 1$

Notation – if $A \in \mathcal{M}_n$ is a matrix, $\mathrm{Sp}(A) = \{\lambda \in \mathbb{C} : A\mathbf{v} = \lambda\mathbf{v}, \mathbf{v} \neq \mathbf{0}\}$ is its **spectrum**, i.e., the set of all its eigenvalues and

- $s(A) = \max\{\mathrm{Re}\,(\lambda),\ \lambda \in \mathrm{Sp}(A)\}$ is its **spectral abscissa**
- $\rho(A) = \max\{|\lambda|, \lambda \in \mathrm{Sp}(A)\}$ is its **spectral radius**

Simulating the system

The `R` package we use for ODE (`deSolve`) can also do discrete-time systems, with very little adaptation..

The function call is then of the form

```
sol <- ode(func = tetanus_Cvjetanovic, y = IC, times = 0:30,
           parms = params, method = "iteration")
```

From the help for `ode`

> *Method "iteration" is special in that here the function* `func` *should return the new value of the state variables rather than the rate of change*

The right hand side

```
tetanus_Cvjetanovic = function(t, y, params) {
  with(as.list(c(y, params)), {
    T = S+L_b+L+I_b+I+R+R_b
    dD = pi_IbD*gamma_b*I_b+pi_ID*gamma*I
    delta_T = dD/T
    dS_b = b*T
    dS = b*(1-lambda_b)*(T-R)+nu*R+nu_b*I+pi_IbS*gamma_b*I_b +
      pi_IS*gamma*I-(lambda+d-delta_T)*S
    dL_b = lambda_b*b*(T-R)-(epsilon_b+d-delta_T)*L_b
    dL = lambda*S-(epsilon+d-delta_T)*L
    dI_b = epsilon_b*L_b-(gamma_b+d-delta_T)*I
    dI = epsilon*L-(gamma+d-delta_T)*I
    dR = pi_IbR*gamma_b*I_b+pi_IR*gamma*I-(nu+d-delta_T)*R
    dR_b = b*R-(nu_b+d-delta_T)*R_b
    list(c(S_b+dS_b,S+dS,L_b+dL_b,L+dL,I_b+dI_b,I+dI,R+dR,R_b+dR_b,D+dD))
  })
}
```

Set parameters

```
params = list()
params$epsilon_b = 0.1667
params$epsilon = 0.125
params$gamma_b = 1/3
params$gamma = 0.0714
params$nu = 0.000274
params$nu_b = 0.005479
params$b = 0.00009889
params$d = 0.0000411

params$pi_IbS = 0.05
params$pi_IS = 0.3
params$pi_IbR = 0.05
params$pi_IR = 0.3
params$pi_IbD = 0.9
params$pi_ID = 0.4

params$lambda_b = 0.1
params$lambda = 0.1
```

A last few things then run

```
IC = c(S_b = 0,
  S = 100000,
  L_b = 0,
  L = 0,
  I_b = 0,
  I = 0,
  R = 0,
  R_b = 0,
  D = 0)
tspan = 0:30
sol <- ode(func = tetanus_Cvjetanovic, y = IC, times = tspan,
      parms = params, method = "iteration")
```

A few remarks about this model

To set λ_b and λ, we need to explore numerically model response

Discrete-time models can be analysed in pretty much the same way as continuous time ones, but this one will be hard: there is no DFFP!

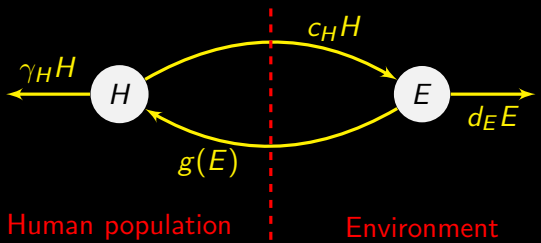
This means the usual methods for computing $\mathcal{R}_0$ will not work, as there is no DFFP to perturb away from...

Some considerations about numerics

- The tetanus model of Cvjetanović
- The model of Capasso for ETP
- The first schistosomiasis model of Woolhouse
- The third schistosomiasis model of Woolhouse – Heterogeneous contacts

Recall the base model of Capasso

$$E' = c_H H - d_E E \tag{4a}$$
$$H' = g(E) - \gamma_H H \tag{4b}$$



$1/\gamma_H$ mean infectious period, $1/d_E$ mean lifetime of the agent in the environment, c_H growth rate of the agent due to the human population, $g(E)$ incidence of the agent on human population

Incidence function

$$g(E) = h(E)N\beta p \tag{5}$$

where

- $h(E)$ probability for an exposed susceptible to get the infection
- N total human population
- β fraction of susceptible individuals in N
- p fraction exposed to contaminated environment per unit time ("probability per unit time to have a "snack" of contaminated food")

Typically, we would assume p and β independent of E and H and h to be saturating. We take a Holling type II functional response

$$h(E) = h_{max}\frac{E}{h_{half} + E} \tag{6}$$

Simulating (in R) – Incidence function

```r
h = function(E, params) {
  # Use Michaelis Menten (Holling type II) growth
  OUT = params$g_max * E / (params$g_half+E)
  return(OUT)
}
g = function(E, params) {
  OUT = params$N * params$beta * params$p * h(E,params)
  return(OUT)
}
```

The right hand side

```
rhs_Capasso_ODE = function(t, x, params) {
  with(as.list(c(x, params)), {
    dE = c_H*H-d_E*E
    dH = g(E, params)-gamma_H*H
    list(c(dE, dH))
  })
}
```

Setting parameters

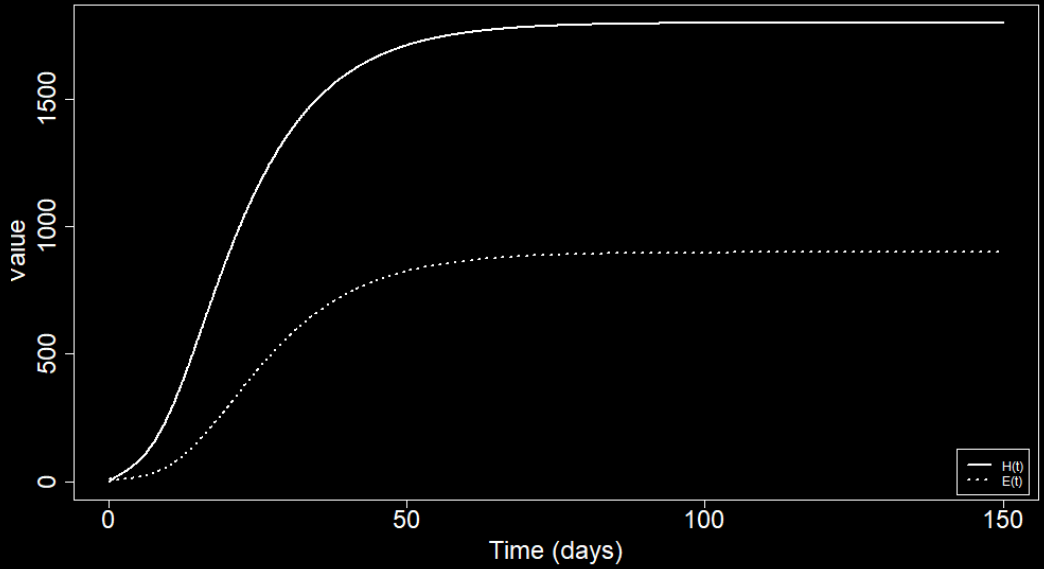
```
# Put parameters in a list
params = list()
params$N = 1000       # Total population
params$gamma_H = 1/10 # Infectious period
params$d_E = 1/5      # Lifetime agent
params$c_H = 0.1      # Flow from humans
# Human characteristics and behaviour
params$beta = 0.2 # Fraction susceptible
params$p = 0.1    # Probability of having "snack"
# Growth function
params$g_max = 10
params$g_half = 100
# Final time
params$t_f = 150
```

Running and plotting (base)

```
IC <- c(E = 10, H = 0)
tspan = seq(from = 0, to = params$t_f, by = 0.1)

sol_ODE = ode(y = IC,
              func = rhs_Capasso_ODE,
              times = tspan,
              parms = params)

plot(sol_ODE[,"time"], sol_ODE[,"H"],
      type = "l", lwd = 2,
      xlab = "Time␣(days)", ylab = "Value")
lines(sol_ODE[,"time"], sol_ODE[,"E"],
      lwd = 2, lty = 3)
legend("bottomright", legend = c("H(t)", "E(t)"),
       lwd = c(2,2), lty = c(1,3), inset = 0.01)
```


Value
0
500
1000
1500
0
50
100
150
Time (days)
H(t)
E(t)

Let

$$\mathcal{R}_0 = \frac{g'_+(0)c_H}{d_E\gamma_H} \tag{7}$$

Theorem 1

- *If* $0 < \mathcal{R}_0 < 1$, *then* (4) *admits only the trivial equilibrium in the positive orthant, which is GAS*
- *If* $\mathcal{R}_0 > 1$, *then two EP exist:* $(0,0)$, *which is unstable, and* $z^\star = (E^\star, H^\star)$ *with* $E^\star, H^\star > 0$, *GAS in* $\mathbb{R}^2_+ \setminus \{0,0\}$

Computing $\mathcal{R}_0$

With the chosen g, we have

$$g'(E) = \frac{N\beta p g_{half} g_{max}}{(g_{half} + E)^2}$$

whence

$$g'_+(0) = \frac{N\beta p g_{max}}{g_{half}}$$

and thus

$$\mathcal{R}_0 = \frac{N\beta p g_{max}}{g_{half}} \frac{c_H}{d_E \gamma_H} \tag{8}$$

```
R0 = function(params) {
  with(as.list(params), {
    R0 = N*beta*p*g_max*c_H / (g_half*d_E*gamma_H)
    return(R0)
  })
}
```

Showing things dynamically using Shiny

Shiny is an `R` library (made by RStudio) to easily make interactive displays

See some documentation here

Some examples here and here

Create a subdirectory with the name of your app and a file called `app.R` in there

Structure of a Shiny app

Need to use library `shiny`

Define two elements

- `ui`, which sets up the user interface
- `server`, which handles the computations, generation of figures, etc.

I explain different elements as we progress. See the code in the `CODE` folder and `Capasso_simpleETP_shiny` subdirectory

The `ui` part

Here, we use `fluidPage` to create the UI. There are other functions: `fillPage`, `fixedPage`, `flowLayout`, `navbarPage`, `sidebarLayout`, `splitLayout` and `verticalLayout`

```
# Define UI
ui <- fluidPage(
)
```

We now fill this function

A title and some sliders

```
# Application title
titlePanel("Simple␣ETP␣model␣of␣Capasso"),
# Sidebar with slider inputs for some parameters
sidebarLayout(
    sidebarPanel(
      sliderInput("inv_gamma_H",
                  "Average␣infectious␣period␣(days):",
                  min = 0,
                  max = 30,
                  value = 10),
      sliderInput("c_H",
                  "Flow␣from␣humans:",
                  min = 0,
                  max = 2,
                  value = 0.1),
```

Plus other sliders for all other parameters

Note the little trick...

```
sliderInput("inv_gamma_H",
"Average␣infectious␣period␣(days):",
min = 0,
max = 30,
value = 10),
```

I want to give a user friendly version of the parameter value, using the number of days rather than the inverse, whereas the model uses the latter. So I prefix the variable name by `inv_` and then process as follows in the `server` part

```
params <- list()
for (param_name in names(input)) {
  if (grepl("inv_", param_name)) {
    new_param_name = gsubs("inv_", "", param_name)
    params[[new_param_name]] = 1/input[[param_name]]
  } else {
    params[[param_name]] = input[[param_name]]
  }
}
```

The simulation functions can be outside of `ui` or `server`, this makes the code neater

These functions are the same as before (right hand side, g, h, R0), so they are not shown here

The `server` part

```
# Define server logic required to draw the result
server <- function(input, output) {
  ##
  ## Expression that generates the plot
  ##
  output$a_odePlot <- renderPlot({
    params <- list()
    params$N = 1000 # We could let this vary, we don't here..
    for (param_name in names(input)) {
      if (grepl("inv_", param_name)) {
        new_param_name = gsub("inv_", "", param_name)
        params[[new_param_name]] = 1/input[[param_name]]
      } else {
        params[[param_name]] = input[[param_name]]
      }
    }
    # Initial conditions and time span
    IC <- c(E = 10, H = 0)
    tspan <- seq(from = 0, to = params$tf, by = 0.1)
```

The `server` part (continued)

```
    # Compute solution
    sol_ODE = ode(y = IC,
                  func = rhs_Capasso_ODE,
                  times = tspan,
                  parms = params)
    # Make the plot
    y_max = max(max(sol_ODE[,"H"]),sol_ODE[,"E"])
    plot(sol_ODE[,"time"], sol_ODE[,"H"],
          type = "l", lwd = 2,
          xlab = "Time␣(days)", ylab = "Value",
          ylim = c(0, y_max),
          main = sprintf("R_0=%1.2f", round(R0(params),2)))
    lines(sol_ODE[,"time"], sol_ODE[,"E"],
          lwd = 2, lty = 3)
    legend("topleft", legend = c("H(t)", "E(t)"),
           lwd = c(2,2), lty = c(1,3), inset = 0.01)
  })
}
```

Finally, run the code

```
# Run the application
shinyApp(ui = ui, server = server)
```

Adding a periodic component

Assume p in (5) takes the form

$$p(t) = p(t+\omega) > 0, \quad t \in \mathbb{R} \tag{9}$$

i.e., p has period ω. So we now consider the incidence

$$g(t, E) = p(t)h(E) \tag{10}$$

with h having the properties prescribed earlier. Letting

$$p_{min} := \min_{0 \le t \le \omega} p(t), \quad p_{max} := \max_{0 \le t \le \omega} p(t) \tag{11}$$

then we require that

$$\lim_{z \to \infty} \frac{g(z)}{z} < \frac{d_E \gamma_H}{c_H p_{max}} \tag{12}$$

Let

$$\mathcal{R}_0^{min} = \frac{c_H p_{min} h'_+(0)}{d_E \gamma_H}, \quad \mathcal{R}_0^{max} = \frac{c_H p_{max} h'_+(0)}{d_E \gamma_H} \tag{13}$$

Theorem 2

- *If* $0 < \mathcal{R}_0^{max} < 1$, *then* (4) *with incidence* (10) *always goes to extinction*
- *If* $\mathcal{R}_0^{min} > 1$, *then a unique nontrivial periodic endemic state exists for* (4) *with incidence* (10)

How to add periodicity in numerics?

```
p_t = function(t, params) {
  angle = 2*pi/params$p_period
  OUT = cos(angle*t) # Make the base cos wave
  OUT = OUT/2*(params$p_max-params$p_min) # Scale
  OUT = OUT-min(OUT)+params$p_min # Shift up
  return(OUT)
}
g = function(E, params, t) {
  OUT = params$N * params$beta * p_t(t, params) * h(E,params)
  return(OUT)
}
R0 = function(params) {
  with(as.list(params), {
    R0 = list()
    R0$min = N*beta*p_min*g_max*c_H / (g_half*d_E*gamma_H)
    R0$max = N*beta*p_max*g_max*c_H / (g_half*d_E*gamma_H)
    return(R0)
  })
}
```

Some considerations about numerics

- The tetanus model of Cvjetanović
- The model of Capasso for ETP
- The first schistosomiasis model of Woolhouse
- The third schistosomiasis model of Woolhouse – Heterogeneous contacts

The model

Population of H individuals using a body of water containing N snails

I_H mean number of schistosomes per person and i_S the proportion of patent infections in snails (prevalence)

$$I_H' = \alpha N i_S - \gamma I_H \tag{14a}$$
$$i_S' = \beta H I_H (1 - i_S) - \mu_2 i_S \tag{14b}$$

- α number of schistosomes produced per person per infected snail per unit time
- $1/\gamma$ average life expectancy of a schistosome
- $1/\mu_2$ average life expectancy of an infected snail
- β transmission parameter

Simulating – The ODE

```r
# Right hand side of the ODE
rhs_Woolhouse1_ODE = function(t, x, params) {
  with(as.list(c(x, params)), {
    dI_H = alpha*N*i_S-gamma*I_H
    di_S = beta*H*I_H*(1-i_S)-mu_2*i_S
    list(c(dI_H, di_S))
  })
}
```

Let the basic reproductive rate for schistosomes be

$$\mathcal{R}_0 = \frac{\alpha N \beta H}{\gamma \mu_2} \tag{15}$$

(14) has two EP

- $(I_H^\star, i_S^\star) = (0, 0)$, LAS when $\mathcal{R}_0 < 1$ and unstable when $\mathcal{R}_0 > 1$
- $(I_H^\star, i_S^\star) = \left(\frac{\alpha N}{\gamma} - \frac{\mu_2}{\beta H}, 1 - \frac{1}{\mathcal{R}_0} \right)$, which only "exists" when $\mathcal{R}_0 > 1$ (and is LAS then)

Using $\mathcal{R}_0$ to set β

```
# Put parameters in a list
params = list()
params$H = 100        # Total human population
params$N = 1000       # Total population snails
params$alpha = 20     # Nb schistosomes/infected H/unit time
params$gamma = 1/1000 # Life expectancy schistosome
params$mu_2 = 1/70    # Life expectancy infected snail
# Set beta through R_0:
# R_0= alpha*N*beta*H/(gamma*mu_2),
# so, given R_0,
# beta = R_0*gamma*mu_2/(alpha*N*H)
params$R_0 = 2.5     # Desired value of R_0
params$beta = params$R_0*params$gamma*params$mu_2 /
  (params$alpha*params$N*params$H)
```

Helping these computations

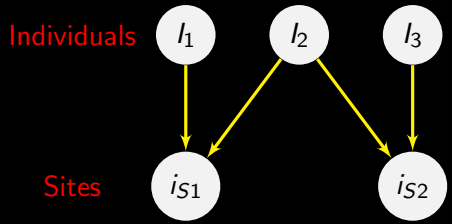
```
R0_Woolhouse_ODE = function(params) {
  with(as.list(params), {
    R0 = alpha*N*beta*H/(gamma*mu_2)
    return(R0)
  })
}
EEP_Woolhouse_ODE = function(params) {
  with(as.list(params), {
    OUT = list()
    OUT$I_H = alpha*N/gamma-mu_2/(beta*H)
    OUT$i_S = 1-1/R0_Woolhouse_ODE(params)
    return(OUT)
  })
}
```

Some considerations about numerics

- The tetanus model of Cvjetanović
- The model of Capasso for ETP
- The first schistosomiasis model of Woolhouse
- The third schistosomiasis model of Woolhouse – Heterogeneous contacts

Heterogeneities in contact rates

I_i the number of schistosomes in person $i = 1, \ldots, H$ and i_{Sj} the proportion of patent infected snails in site $j = 1, \ldots, L$ (L sites each supporting N snails)



I_i the number of schistosomes in person $i = 1, \ldots, H$ and i_{Sj} the proportion of patent infected snails in site $j = 1, \ldots, L$ (L sites each supporting N snails)

$$I_i' = \alpha \left(\sum_j \eta_{ij} N i_{Sj} \right) - \gamma I_i \tag{16a}$$

$$i_{Sj}' = \beta \left(\sum_i \eta_{ij} I_i \right) (1 - i_{Sj}) - \mu_2 i_{Sj} \tag{16b}$$

η_{ij} rate of water contact by individual i at site j

How to deal with large systems

A system like (16) is a **large system** of ODE, as there are $H + L$ differential equations, with that number potentially large

Large systems of this type (very few different types of equations) are quite simple numerically but require some organisation...

Rather than name the state variables, it is better to use the vector $\mathbf{x}$ with indices for the different types of variables

Indexing positions

```
params = list()
params$H = 100        # Total human population
params$L = 5          # Number of sites
```

Then if we define

```
params$idx_I_H = 1:params$H
params$idx_i_S = (params$H+1):(params$H+params$L)
```

in the ODE, we will be able to use something like

```
I_H = x[params$idx_I_H]
i_S = x[params$idx_i_S]
```

Computing the incidence

Here again, easy to do (and computationally efficacious) provided you are careful

$K = [\eta_{ij}]$ is an $H \times L$ matrix. Denote $I_H = (I_1, \ldots, I_H)^T$ and $i_S = (i_{S1}, \ldots, i_{SL})^T$

Then

$$\sum_j \eta_{ij} N i_{Sj} = N \sum_j \eta_{ij} i_{Sj} = K i_S$$

and

$$\sum_i \eta_{ij} I_i = I_H^T K$$